# Applying Virtual Reality

**by John Locke**
**Code CS, Naval Postgraduate School**
**833 Dyer Road**
**Monterey, CA 93943-5118**
**408/656-2844, 408/656-2814 (fax)**

## The emerging field of Virtual Reality

Advancements in computer processing speed, graphics, networks, and specialized peripherals have allowed the creation of Virtual Reality (VR) applications which immerse a user within a simulated three-dimensional environment. Still in infancy, much remains to be done before the potential of VR is achieved, but the field shows great promise.

In a full VR implementation, the user wears specialized goggles that substitute small video displays for the lenses, one for each eye. The displays become the "lenses" onto a simulated world. The head-mounted display containing the goggles detects head movement and updates the user's view accordingly; the user has the impression of looking around, even moving within, the simulated environment. The sensory experience can be further enriched with sound, or a data glove to simulate contact between the user and simulated objects. In a partial VR implementation, a computer's video display becomes a window onto the simulated environment, and no external gear is required. In a networked environment, people can be immersed in the same simulation, each perceiving it from their own point of view.

## Applying VR to real-world problems

VR applications become practical if creating a simulated environment is cheaper than placing the user in an actual physical environment, or if the physical environment is otherwise inaccessible. For example, an architect can simulate a dwelling allowing the future occupant to explore the interior, assessing the placement of windows, doors, appliances, and to "edit" the dwelling to the occupant's preference.

In the Computer Science Department of the Naval Postgraduate School (NPS), we have developed a VR implementation called NPSNET. As a test-bed for VR research, NPSNET incorporates many generally applicable ideas. However, the application is oriented toward the needs of our military sponsors.

Military environments may be inaccessible for many reasons. A potential area of operations may be strategically or politically off limits. Since the shape of the environment may affect the conduct of operations, simulating significant geographical and man-made features provides a dynamic tool for preparing personnel, essentially a three-dimensional map that can be explored in the user's choice of scale. Furthermore, operations can be rehearsed within the environment with workstations substituting for user-driven entities like tanks or aircraft. The computer can simulate entities as needed to enhance realism, or provide an enemy force. A single user can interact with the environment; a networked simulation allows commanders to rehearse coordination on a grand scale. Indeed, multiple levels of activity can simultaneously take place, as in a real-world environment.

Simulators also reduce training expenses, a powerful motive in this era of budgetary constraints. It is extremely expensive--in fuel, maintenance, and other costs--to operate ships, aircraft, and even tanks. Simulators provide a low-cost substitute for some phases of training. For example, simulators acquaint pilots with cockpits and tank drivers with tank controls in mock-ups that substitute VR displays for windows.

## Constructing a virtual world

The primary structural problem is the geometric modeling of the world. The principles of Computer Graphics form the technical foundation. The computer must maintain detailed knowledge of a 3-D world, to display it from any point of view as a 2-D image. Animation results from moving objects within the image (or the point of view) and updating the display.

Graphics programming consists of coding the geometry to model the virtual world in three-space. Objects must be scaled to show the proper perspective from the point of view. The display must conceal hidden surfaces. “Collision detection” must be explicitly coded, else moving objects which make contact would behave as phantoms, passing through each other. We use Silicon Graphics workstations which implement fundamental graphics algorithms in hardware for optimal speed, and provide graphics code libraries for higher-level functions.

Terrain is the floor of the virtual world. We employ a terrain skin, a database of the elevations at regular grid points (posts). Each set of four adjacent posts is divided into two triangles, each triangle a 2-D polygon to be rendered. It’s a skin because only the surface of the earth need be displayed--the underlying ground is hidden. The terrain skin database also contains zero-height features like lakes, rivers, or roads, and encodes ground types--dirt, grass, field--to allow realistic rendering. A second database contains static objects with height--trees, buildings, telephone poles. Relevant portions of the databases are merged at render-time.

Where do the databases come from? Databases corresponding to actual terrains have been created for other purposes by organizations like the Defense Mapping Agency. Techniques for building the databases range from converting satellite photographs to physically measuring terrain. There is no standard method for encoding--resolution (post spacing) and richness of detail varies, as does the arrangement of the data. NPSNET was designed to recognize two common formats; others must be converted.

We assume that terrain is static, not dynamic, but this limitation will be addressed in the future. The changing nature of terrain due to bomb craters forming, bridges falling, or environmental effects, significantly affects the course of battle. A robust simulation must alter the terrain without permanently changing the original database. In a networked environment, terrain changes must be shared between simulators.

A third database utilized by NPSNET tracks dynamic entities, the vehicles or aircraft that operate in the simulation. In practice, a few entities will be near the user with most farther away. A composite display strategy dictates a detailed model for near entities, a simplified model for entities too distant to make their detail meaningful to the eye.

Entity motion is limited by the laws of physics and the specific entity's dynamics--speed, maneuverability, and other operating parameters. As much as practical, the entity model follows its real-life equivalent. An entity's movement can be input from various sources. If a single simulator is used as a tank or aircraft trainer, the entity is controlled from the keyboard and joystick, or from a specialized input device like a flight stick. The display is an out-the-window view with, optionally, sub-divided sections for instruments, radar screen, or 2-D overhead map. NPSNET also generates and moves virtual entities allowing the operator to interact. An automated entity can obey some mechanistic algorithm; ideally, Artificial Intelligence provides lesser predictability as in the Modular Semi-Automated Forces (ModSAF) system which generates quite realistic entities.

Apart from the out-the-window view we associate with flight simulators, the operator can control the entity from without. Or the operator can "tether" to an entity to transparently ride along. In fact, the operator need not be associated with an entity and can freely explore in what we call a "stealth" mode.

## Environmental effects

The framework of the simulation is a realistic environment with moving entities. But this is a schematic, excessively orderly representation of the physical world. Many environmental factors affect perception and the behavior of entities. The consequences of weather make a big impact. Fog or rain seriously affects visibility. The muddying of the terrain, or other deleterious effects of moisture, severely curtails troop efficiency and equipment reliability. Dust, from natural or man-made causes, affects visibility. Sunlight varies with time of day; moonlight with time of month. Additionally, the combination of light and weather creates widely variable composite conditions, including the patchwork of shadows caused by clouds. Others environmental factors include the effect of wind on aircraft flight and ballistics, and the effect of tides and oceanographic conditions on nautical operations.

A factor that influences the user's sense of immersion is sound. In the real world, we take the omnipresence of sounds for granted, thus a silent virtual environment seems eerily alien. To enhance NPSNET, we added "spatial audio," a utility that surrounds the user with the sounds--typically, entity movements or explosions--that fall with hypothetical earshot of the user's location in the virtual environment. Four speakers surround the user, on the corners of a square. The intensity of a sound is determined by the distance from the user to the sound source in the virtual environment; the spatial location is derived from an algorithm that distributes the sound among subsets of the speakers. The sound itself is transmitted as MIDI data from the simulator to a sampler which plays pre-recorded patterns.

## Networking simulators

Simulations achieve greater utility distributed among machines on a network. A single machine makes sense as a vehicle trainer, or as a way for an operator to engage computer-driven entities.

But a broader goal is to allow human interaction, to engage others through individual control of entities, or to develop command-and-control structures within teams.

The overriding issue is maintaining world consistency. How do separate machines present the same virtual world to their users? An initial state, consisting of the terrain and entity models, is replicated at each workstation before the simulation begins. Once underway, changes to the scenario must be communicated across the network, requiring a protocol that carries the data in a recognizable format. We built NPSNET with the Distributed Interactive Simulation (DIS) protocols, a current standard for interconnecting simulators. DIS defines packet formats for changes in entity state (speed, location, appearance), firing and impact of projectiles, radar emissions, and other capabilities. DIS also defines the rules under which packets are sent; for instance, an entity must announce its state within five second intervals or be inactivated from the scenario.

While DIS has been useful for in-house development, the standard has the greater benefit of allowing interoperability between diverse simulators. If separate organizations develop DIS-compliant flight simulators, they can run on the same network with each seeing the other's aircraft within its scenario. Other simulators specialized for ground vehicles, watercraft, or computer-generated entities can also "plug in." NPSNET in "stealth" mode can view the action. Naturally, the real world doesn't work that easily. DIS is sufficiently broad that no simulator implements the entire standard but, rather, some useful subset. Therefore, interoperability also depends on simulators implementing enough common elements.

## Interactions between the real and simulated worlds

In early 1994, NPS performed an experiment with the U.S. Army's Training Experimentation Command (TEC) at Ft. Hunter Liggett in California's coastal range. TEC field tests military equipment in the hilly terrain using a variety of vehicles, aircraft, and personnel. Their persistent problems have been the inability of an observer to see the entire area of operations, and the difficulty in reviewing the completed exercise in broad scale. The exercises do create a digital record, though. A set of antennas mounted on hilltops poll vehicles through radio transmission for location data, allowing vehicle location to be triangulated. For the joint exercise, NPSNET used a terrain database corresponding to the area of operations. Location data was transmitted from vehicle to antenna to computer to an NPS receiver, converted to DIS packets, and broadcast for input to NPSNET. The result was a real-time 3-D display of a live exercise. NPSNET's logging facility allowed the exercise to be saved and replayed in NPSNET where it could be studied at leisure.

The experiment was useful in and of itself, but it also has broad implications for VR technology. It demonstrates that there need be no hard line between the real and simulated worlds. The real world can generate input for the simulated world creating richer, less deterministic scenarios. When the two are mingled, real world entity performance can be compared with simulated entity performance--as in the Turing test, the two should be indistinguishable. VR creates a training and learning environment that would be impractical in the physical world. Conversely, the simulated world could generate input for the real world. Imagine an airborne pilot tracking other aircraft through instrument readings. Some, perhaps all, of those aircraft are VR entities. It doesn't matter--if the scenario taxes the pilot with a realistically complex problem.

## Performance Issues

Virtual Reality promises great practical benefits. That’s the good news. The bad news is that technology trails our ambitions. Processors, memory and networks are a finite resource and, like gases, applications always seem to expand to fill the available space, forcing system designers to balance competing goals.

Graphics performance, for instance, is at odds with real-time operation. The image refresh rate determines the smoothness of the animation, a human factors consideration. We attempt to limit all-around computational load to allow fifteen refreshes per second. Below that, movement appears jerky and difficult to watch. But we can’t slow the simulation below real-time to enforce a smooth display. As a general strategy, NPSNET limits the polygon count (graphics performance is typically measured by polygons/second rendered). Polygons per entity becomes critical as the number of entities in view climbs into the hundreds or even thousands, thus the average entity consists of only seventeen polygons. Various algorithms are employed such as simplifying the image as distance from the view point increases.

Incorporating environmental effects into the simulator introduces a trade-off between realism and computational load. Increased realism limits the moving entities that can be tracked by the system while still maintaining real-time throughput.

A near absolute degree of realism is unobtainable with current systems. Consider simulating clouds. They have different shapes, sizes, and degrees of opacity, all of which change over time. They drift with the wind and cast shadows. The dynamics are too complicated to be animated in real-time at any refresh rate. Currently, NPSNET gets by with clusters of geometric “puffs” which combine in partially-random patterns. The look is cartoonish but, in practice, users quickly acclimate to sparse detail.

Network bandwidth has lagged behind the logarithmic increases in capacity that other media have enjoyed. Thus, while simulators must share state changes, there is a strong incentive to minimize the communication overhead. This need is reflected in the DIS protocol. State changes are not issued continually by an entity in constant motion. Instead, significant changes (in speed or direction) are issued, but between times, simulators “dead reckon” entities that are under the control of other simulators. Simulators employ a common algorithm to plot where the entity should be, given its last-known speed and direction. A simulator also dead reckons its own entities to measure the difference between the actual and dead reckoned positions. When the difference exceeds a set threshold, a state change is issued to update the other simulators. But rather than “jumping” the entity back to its real location--a disconcerting visual effect--the simulator employs a smoothing algorithm to transition the entity back. The net effect of dead reckoning and smoothing is a transfer of resource utilization from network to processor which, like other conflicts, must be balanced for optimal benefit.

## Conclusion

By now, it should be apparent that we have lost the luxury of specializing in our first field, computer science. In order to simulate real world conditions, we have been forced to become physicists, meteorologists, and oceanographers, at least well enough to translate environmental and physical effects into efficient, but meaningful, mathematical models that dovetail with our graphics algorithms.

Ultimately, the gauge of success is not technology for its own sake, but the application of technology for useful purposes. If some of our expectations sound like pie-in-the-sky, it's because our work is largely experimental at this time. The real payoffs are expected in ten or more years when maturing VR techniques merge with superior hardware to allow significantly more complex and realistic simulations. In the meantime, there is much to be discovered and a real world of opportunity.

## Acknowledgments

The author would like to thank NPS Professor of Computer Science, David R. Pratt, who designed and coded NPSNET, solving a lot of tricky problems along the way; Professor Michael Zyda, whose idea it was in the first place; Systems Programmer, Paul Barham, who contributed many innovative ideas; our dedicated students who have extended NPSNET in ways we never imagined; and, of course, our sponsors, DMSO, ARL, ARPA, and STRICOM.

## Read more about it

Durlach, Nathaniel I., and Mavor, Anne S., Editors, *Virtual Reality: Scientific and Technological Challenges*, National Academy Press, Washington, D.C., 1994

Institute for Simulation and Training, *Military Standard: Protocol Data Units for Entity Information and Entity Interaction in a Distributed Interactive Simulation*, IST, Orlando, Florida, October 1991

## About the author

John Locke (jxxl@cs.nps.navy.mil) has been a UNIX systems programmer in the Computer Science Department of NPS for seven years. He has specialized in network programming and wrote the DIS networking library utilized by NPSNET. He has a B.A. in English from the University of California at Berkeley, 1977.